

Simultaneous Learning of Trees and Representations for Extreme Classification, with Application to Language Modeling

Yacine Jernite
jernite@cs.nyu.edu

Anna Choromanska
achoroma@cims.nyu.edu

David Sontag
dsontag@cs.nyu.edu

Yann LeCun
yann@cs.nyu.edu

Courant Institute of Mathematical Sciences, NYU¹

Abstract

This paper addresses the problem of multi-class classification with an extremely large number of classes, where the class predictor is learned jointly with the data representation, as is the case in language modeling problems. The predictor admits a hierarchical structure, which allows for efficient handling of settings that deal with a very large number of labels. The predictive power of the model however can heavily depend on the structure of the tree. We address this problem with an algorithm for tree construction and training that is based on a new objective function which favors balanced and easily-separable node partitions. We describe theoretical properties of this objective function and show that it gives rise to a boosting algorithm for which we provide a bound on classification error, i.e. we show that if the objective is weakly optimized in the internal nodes of the tree, then our algorithm will amplify this weak advantage to build a tree achieving any desired level of accuracy. We apply the algorithm to the task of language modeling by re-framing conditional density estimation as a variant of the hierarchical classification problem. We empirically demonstrate on text data that the proposed approach leads to high-quality trees in terms of perplexity and computational running time compared to its non-hierarchical counterpart.

space. From extreme multi-class classification to language modeling, one commonly used approach to this problem reduces it to a series of choices in a tree-structured model, where the leaves typically correspond to labels. While this allows for faster prediction, and is in many cases necessary to make the models tractable, the performance of the system can depend significantly on the structure of the tree used, e.g. [MH09].

Instead of relying on possibly costly heuristics [MH09], extrinsic hierarchies [MB05] which can badly generalize across different data sets, or purely random trees, we provide an efficient data-dependent algorithm for tree construction and training. Inspired by the LOM tree algorithm [CL15] for binary trees, we present an objective function which favors high-quality node splits, i.e. balanced and easily separable. In contrast to previous work, our objective applies to trees of arbitrary width and leads to guarantees on model accuracy. Furthermore, we show how to successfully optimize it in the setting when the data representation needs to be learned simultaneously with the classification tree.

Finally, the multi-class classification problem is closely related to that of conditional density estimation [RG11, Bis06] since both need to consider all labels (at least implicitly) during learning and at prediction time. Both problems present similar difficulties when dealing with very large label spaces, and the techniques that we present in this work can be applied indiscriminately to either. Indeed, we show how to adapt our algorithm to efficiently solve the conditional density estimation problem of learning a tree-structured language model.

This paper is organized as follows: Section 2 discusses related work, Section 3 outlines the necessary background and defines the flat and tree-structured objectives for multi-class classification and density estimation, Section 4 presents the objective and the optimization algorithm, Section 5 contains theoretical results, Section 6 adapts the algorithm to the problem of language modeling, Section 7 reports empirical re-

1 Introduction

Several machine learning settings are concerned with performing predictions in a very large discrete label

¹Yann LeCun: and Facebook AI Research

sults on the Penn TreeBank and Gutenberg corpora, and finally Section 8 concludes the paper. Supplementary material contains additional material and proofs of theoretical statements of the paper. We also release the implementation of our algorithm, which is done in Torch [CKF11].

2 Related Work

The multi-class classification problem has been addressed in the literature in a variety of ways. Some examples include i) clustering methods [BWG10, MGC09, WMY13] ([BWG10] was later improved in [DSBFF11]), ii) sparse output coding [ZX13], iii) variants of error correcting output codes [HKLZ09], iv) variants of iterative least-squares [AKK⁺14], v) a method based on guess-averse loss functions [BSKV14], and vi) classification trees [BLR09, CL15, DKLM16] (that includes the Conditional Probability Trees [BLL⁺09] when extended to the multi-class classification setting).

The recently proposed LOM tree algorithm [CL15] differs significantly from other similar hierarchical approaches, like for example Filter Trees [BLR09] or random trees [Bre01], in that it addresses the problem of learning good-quality binary node partitions. The method results in low-entropy trees and instead of using an inefficient enumerate-and-test approach, see e.g. [BFOS84], to find a good partition or expensive brute-force optimization [AGPV13], it searches the space of all possible partitions with SGD [Bot98]. Another work [DKLM16] uses a binary tree to map an example to a small subset of candidate labels and makes a final prediction via a more tractable one-against-all classifier, where this subset is identified with the proposed Recall Tree. Some other notable approaches based on decision trees include FastXML [PV14] (and its slower and less accurate at prediction predecessor [AGPV13]). The approach is based on optimizing the rank-sensitive loss function and shows an advantage over some other ranking and NLP-based techniques in the context of multi-label classification. Other related approaches include the SLEEC classifier [BJK⁺15] for extreme multi-label classification that learns embeddings which preserve pairwise distances between only the nearest label vectors and ranking approaches based on negative sampling [WBU11]. Another tree approach [KFCB15] shows no computational speed up but leads to significant improvements in prediction accuracy.

Conditional density estimation can also be challenging in settings where the label search is large, and one such case is that of language modeling. The task of language modeling consists in learning a probability density function over word sequences. Language models learn word representations that are useful in a number of practical settings such as named entity

recognition [TRB10], parsing [SLNM11], semantic role labeling [CW08], sentiment analysis [MN10], and document retrieval and classification [DWR⁺15]. One common approach to the problem consists in using the chain rule to reduce the problem to learning the probability of a word given the previous text, either by making a Markov assumption and approximating the left context by the last few words seen (n-grams e.g. [JM80, Kat87], feed-forward neural language models [MT12, MDK⁺11, SG02]), or by attempting to learn a low-dimensional representation of the full history (RNNs [MKB⁺10, MV15, TSM15, KIS⁺15]). Both the recurrent and feed-forward Neural Probabilistic Language Models (NPLM) [BDVJ03a] simultaneously learn a distributed representation for words and the probability function for word sequences, expressed in terms of these representations.

The major drawback of these models is that they can be slow to train, as they grow linearly with the vocabulary size (anywhere between 10,000 and 1M words), which can make them difficult to apply [MT12]. A number of methods have been proposed to overcome this difficulty. Works such as LBL [MH07] or Word2Vec [MSC⁺13] reduce the model to its barest bones, with only one hidden layer and no non-linearities. Another proposed approach has been to only compute the NPLM probabilities for a reduced vocabulary size, and use hybrid neural-*n*-gram model [SG05] at prediction time. Other avenues to reduce the cost of computing gradients for large vocabularies include using different sampling techniques to approximate it [BS03, BS08, MT12], replacing the likelihood objective by a contrastive one [GH12] or spherical loss [dBV16], relying on self-normalizing models [AK15], taking advantage of data sparsity [VdBB15], or using clustering-based methods [GJC⁺16].

Similarly to the classification case, there have also been a significant number of works that use tree structured models to accelerate computation of the likelihood and gradients [MB05, MH09, DWR⁺15, MSC⁺13]. These use various heuristics to build a hierarchy, from using ontologies [MB05] to Huffman coding [MSC⁺13]. One algorithm which endeavors to learn a binary tree structure along with the representation is presented in [MH09]. They iteratively learn word representations given a fixed tree structure, and use a criterion that trades off between making a balanced tree and clustering the words based on their current embedding. The application we present in the second part of our paper is most closely related to the latter work, and uses a similar embedding of the context. However, where their setting is limited to binary trees, we work with arbitrary width, and provide a tree building objective which is both less computationally costly and comes with theoretical guarantees.

3 Background

In this section, we define the classification and log-likelihood objectives we wish to maximize. Let \mathcal{X} be an input space, and \mathcal{V} a label space. Let \mathcal{P} be a joint distribution over samples in $(\mathcal{X}, \mathcal{V})$, and let $f_\Theta : \mathcal{X} \rightarrow \mathbb{R}^{d_r}$ be a function mapping every input $x \in \mathcal{X}$ to a representation $\mathbf{r} \in \mathbb{R}^{d_r}$, and parametrized by Θ (e.g. as a neural network).

We consider two objectives. Let g be a function that takes an input representation $\mathbf{r} \in \mathbb{R}^{d_r}$, and predicts for it a label $g(\mathbf{r}) \in \mathcal{V}$. The classification objective is defined as the expected proportion of correctly classified examples:

$$\mathcal{O}^{\text{class}}(\Theta, g) = \mathbb{E}_{(x,y) \sim \mathcal{P}} [\mathbb{1}[g \circ f_\Theta(x) = y]] \quad (1)$$

Now, let $p_\theta(\cdot|\mathbf{r})$ define a conditional probability distribution (parametrized by θ) over \mathcal{V} for any $\mathbf{r} \in \mathbb{R}^{d_r}$. The density estimation task consists in maximizing the expected log-likelihood of samples from $(\mathcal{X}, \mathcal{V})$:

$$\mathcal{O}^{\text{ll}}(\Theta, \theta) = \mathbb{E}_{(x,y) \sim \mathcal{P}} [\log p_\theta(y|f_\Theta(x))] \quad (2)$$

Tree-Structured Classification and Density Estimation Let us now show how to express the objectives in Equations 1 and 2 when using tree-structured prediction functions (with fixed structure) as illustrated in Figure 1.

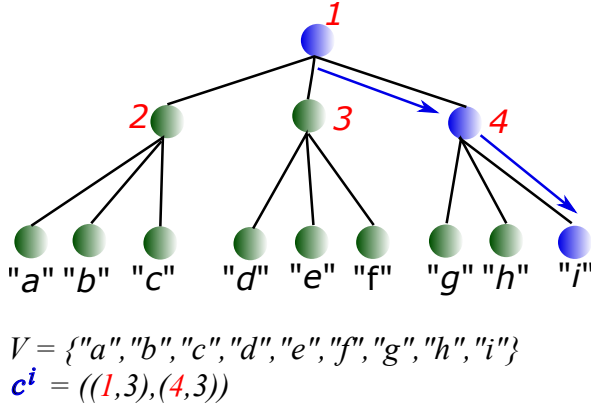


Figure 1: Hierarchical predictor: in order to predict label “i”, the system needs to choose the third child of node 1, then the third child of node 4.

Consider a tree \mathcal{T} of depth D and arity M with $K = |\mathcal{V}|$ leaf nodes and N internal nodes. Each leaf l corresponds to a label, and can be identified with the path \mathbf{c}^l from the root to the leaf. In the rest of the paper, we will use the following notations:

$$\mathbf{c}^l = ((c_{1,1}^l, c_{1,2}^l), \dots, (c_{d,1}^l, c_{d,2}^l), \dots, (c_{D,1}^l, c_{D,2}^l)), \quad (3)$$

where $c_{d,1}^l \in [1, N]$ correspond to the node index at depth d , and $c_{d,2}^l \in [1, M]$ indicates which child of $c_{d,1}^l$

is next in the path. In that case, our classification and density estimation problems are reduced to choosing the right child of a node or defining a probability distribution over children given $x \in \mathcal{X}$ respectively.

We then need to replace g and p_θ with node decision functions $(g_n)_{n=1}^N$ and conditional probability distributions $(p_{\theta_n})_{n=1}^N$ respectively. Given such a tree and representation function, our objective functions then become:

$$\mathcal{O}^{\text{class}}(\Theta, g) = \mathbb{E}_{(x,y) \sim \mathcal{P}} \left[\prod_{d=1}^D \mathbb{1}[g_{c_{d,1}^l} \circ f_\Theta(x) = c_{d,2}^l] \right] \quad (4)$$

$$\mathcal{O}^{\text{ll}}(\Theta, \theta) = \mathbb{E}_{(x,y) \sim \mathcal{P}} \left[\sum_{d=1}^D \log p_{\theta_{c_{d,1}^l}}(c_{d,2}^l | f_\Theta(x)) \right] \quad (5)$$

The tree structured objectives defined above in Equations 4 and 5 can be optimized in the space of parameters of the representation and node functions using standard gradient ascent methods. However, they also implicitly depend on the tree structure \mathcal{T} . In the rest of the paper, we provide a surrogate objective function which determines the structure of the tree and, as we show theoretically (Section 5), maximizes the criterion in Equation 4 and, as we show empirically (Sections 6 and 7), maximizes the criterion in Equation 5.

4 Learning Tree-Structured Objectives

In this section, we introduce a per-node objective J_n which leads to good quality trees when maximized, and provide an algorithm to optimize it.

4.1 Objective function

We define the node objective J_n for node n as:

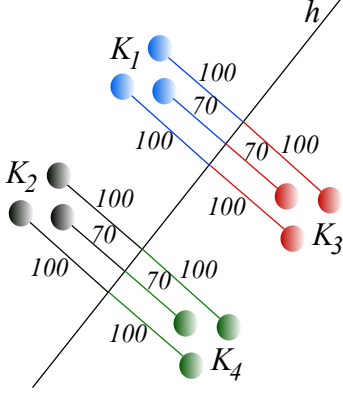
$$J_n = \frac{2}{M} \sum_{i=1}^K q_i^{(n)} \sum_{j=1}^M |p_j^{(n)} - p_{j|i}^{(n)}|, \quad (6)$$

where $q_i^{(n)}$ denotes the proportion of nodes reaching node n that are of class i , $p_{j|i}^{(n)}$ is the probability that an example of class i reaching n will be sent to its j^{th} child, and $p_j^{(n)}$ is the probability that an example of any class reaching n will be sent to its j^{th} child. Note that with this notation, we have:

$$\forall j \in [1, M], \quad p_j^{(n)} = \sum_{i=1}^K q_i^{(n)} p_{j|i}^{(n)} \quad (7)$$

The objective in Equation 6 reduces to the LOM tree objective in the case of $M = 2$.

At a high level, maximizing the objective encourages the conditional distribution for each class to be as different as possible from the global one; so the node



Discrete:
 $p_1^{(n)} = \frac{6}{12} = \mathbf{0.5}$

$$p_{1|1}^{(n)} = \frac{3}{3} = \mathbf{1}, \quad p_{1|2}^{(n)} = \frac{3}{3} = \mathbf{1}, \quad p_{1|3}^{(n)} = \frac{0}{3} = \mathbf{0}, \quad p_{1|4}^{(n)} = \frac{0}{3} = \mathbf{0}$$

Continuous:

$$p_1^{(n)} = \frac{1}{12}(\sigma(100) + \sigma(70) + \dots + \sigma(-70) + \sigma(-100)) \approx \mathbf{0.5}$$

$$p_{1|1}^{(n)} = \frac{1}{3}(\sigma(100) + \sigma(70) + \sigma(100)) \approx \mathbf{1}$$

$$p_{1|2}^{(n)} = \frac{1}{3}(\sigma(100) + \sigma(70) + \sigma(100)) \approx \mathbf{1}$$

$$p_{1|3}^{(n)} = \frac{1}{3}(\sigma(-100) + \sigma(-70) + \sigma(-100)) \approx \mathbf{0}$$

$$p_{1|4}^{(n)} = \frac{1}{3}(\sigma(-100) + \sigma(-70) + \sigma(-100)) \approx \mathbf{0}$$

Figure 2: The comparison of discrete and continuous definitions of probabilities $p_j^{(n)}$ and $p_{j|i}^{(n)}$ on a simple example with $K = 4$ classes and binary tree ($M = 2$). n is an exemplary node, e.g. root. σ denotes sigmoid function. Color circles denote data points.

decision function needs to be able to discriminate between examples of the different classes. The objective thus favors balanced and pure node splits. To wit, we call a split at node n *perfectly balanced* when the global distribution $p^{(n)}$ is uniform, and *perfectly pure* when each $p_{j|i}^{(n)}$ takes value either 0 or 1, as all data points from the same class reaching node n are sent to the same child.

In Section 5 we discuss the theoretical properties of this objective in details. We show that maximizing it leads to perfectly balanced and perfectly pure splits. We also derive the boosting theorem that shows the number of internal nodes that the tree needs to have to reduce the classification error below any arbitrary threshold, under the assumption that the objective is “weakly” optimized in each node of the tree.

Remark 1. *In the rest of the paper, we use node functions g_n which take as input a data representation $\mathbf{r} \in \mathbb{R}^d$ and output a distribution over children of n (for example using a soft-max function). When used in the classification setting, g_n sends the data point to the child with the highest predicted probability. With this notation, and representation function f_Θ , we can write:*

$$p_j^{(n)} := \mathbb{E}_{(x,y) \sim \mathcal{P}}[g_n \circ f_\Theta(x)] \quad (8)$$

and

$$p_{j|i}^{(n)} := \mathbb{E}_{(x,y) \sim \mathcal{P}}[g_n \circ f_\Theta(x) | y = i] \quad (9)$$

One could define $p_j^{(n)}$ as the ratio of the number of examples that reach node n and are sent to its j^{th} child to the total the number of examples that reach node n and $p_{j|i}^{(n)}$ as the ratio of the number of examples that reach node n , correspond to label i , and are sent to the j^{th} child of node n to the total the number of examples that reach node n and correspond to label i . We instead look at the continuous counter-parts of these discrete definitions as given by Equations 8 and 9 and illustrated in Figure 2 (note that continuous defi-

nitions have elegant geometric interpretation based on margins), which simplifies the optimization problem.

Algorithm 1 Tree Learning Algorithm

Input Input representation function: f with parameters Θ_f . Node decisions functions $(g_n)_{n=1}^K$ with parameters $(\Theta_n)_{n=1}^K$. Gradient step size ϵ .

Output Learned M -ary tree of depth D , parameters Θ_f and $(\Theta_n)_{n=1}^K$.

```

1: procedure INITIALIZE_NODE_STATS()
2:   for  $n = 1$  to  $N$  do
3:     for  $i = 1$  to  $K$  do
4:       SumProbas $_{n,i} \leftarrow \mathbf{0}$ 
5:       Counts $_{n,i} \leftarrow \mathbf{0}$ 
6: procedure NODE_COMPUTE( $\mathbf{w}, n, i, \text{target}$ )
7:    $\mathbf{p} \leftarrow g_n(\mathbf{w})$ 
8:   SumProbas $_{n,i} \leftarrow \text{SumProbas}_{n,i} + \mathbf{p}$ 
9:   Counts $_{n,i} \leftarrow \text{Counts}_{n,i} + 1$ 
10:  // Gradient step in the node parameters
11:   $\Theta_n \leftarrow \Theta_n + \epsilon \frac{\partial p_{\text{target}}}{\partial \Theta_n}$ 
12:  return  $\frac{\partial p_{\text{target}}}{\partial \mathbf{w}}$ 

13: INITIALIZE_NODE_STATS()
14: for Each batch  $b$  do
15:  // ASSIGN_LABELS() re-builds the tree based on
  the current statistics
16:  ASSIGN_LABELS( $\{1, \dots, K\}$ , root, 1)
17:  for each example  $(\mathbf{x}, i)$  in  $b$  do
18:    Compute input representation  $\mathbf{w} = f(\mathbf{x})$ 
19:     $\Delta \mathbf{w} \leftarrow \mathbf{0}$ 
20:    for  $d = 1$  to  $D$  do
21:      Set node id and target:  $(n, j) \leftarrow c_d^i$ 
22:       $\Delta \mathbf{w} \leftarrow \Delta \mathbf{w} + \text{NODE\_COMPUTE}(\mathbf{w}, n, j)$ 

23:  // Gradient step in the parameters of  $f$ 
24:   $\Theta_f \leftarrow \Theta_f + \epsilon \frac{\partial f}{\partial \Theta_f} \Delta \mathbf{w}$ 

```

Algorithm 2 Label Assignment Algorithm**Input** Node statistics, max depth D

- 1: Paths from root to labels: $\mathcal{P} = (\mathbf{c}^i)_{i=1}^K$
- 2: node ID n and depth d
- 3: List of labels currently reaching the node

Output Updated paths

- 4: Lists of labels now assigned to each of n 's children under depth constraints

```

5: procedure ASSIGNLABELS(labels,  $n$ ,  $d$ )
6:   // first, compute  $p_j^{(n)}$  and  $p_{j|i}^{(n)}$ .  $\odot$  is the
   element-wise multiplication
7:    $\mathbf{p}_0^{avg} \leftarrow \mathbf{0}$ 
8:   count  $\leftarrow 0$ 
9:   for  $i$  in labels do
10:     $\mathbf{p}_0^{avg} \leftarrow \mathbf{p}_0^{avg} + \text{SumProbas}_{n,i}$ 
11:    count  $\leftarrow$  count +  $\text{Counts}_{n,i}$ 
12:     $\mathbf{p}_i^{avg} \leftarrow \text{SumProbas}_{n,i} / \text{Counts}_{n,i}$ 
13:    $\mathbf{p}_0^{avg} \leftarrow \mathbf{p}_0^{avg} / \text{count}$ 
14:   // then, assign each label to a child of  $n$  under
   depth constraints
15:   unassigned  $\leftarrow$  labels
16:   full  $\leftarrow \emptyset$ 
17:   for  $j = 1$  to  $M$  do
18:     assigned $_j \leftarrow \emptyset$ 
19:   while unassigned  $\neq \emptyset$  do
20:     //  $\frac{\partial J_n}{\partial p_{j|i}^{(n)}}$  is given in Equation 10
21:      $(i^*, j^*) \leftarrow \underset{i \in \text{unassigned}, j \notin \text{full}}{\text{argmax}} \left( \frac{\partial J_n}{\partial p_{j|i}^{(n)}} \right)$ 
22:      $\mathbf{c}_d^l \leftarrow (n, j^*)$ 
23:     assigned $_{j^*} \leftarrow$  assigned $_{j^*} \cup \{i^*\}$ 
24:     unassigned  $\leftarrow$  unassigned  $\setminus \{i^*\}$ 
25:     if |assigned $_{j^*}| =  $M^{D-d+1}$  then
26:       full  $\leftarrow$  full  $\cup \{j^*\}$ 
27:   for  $j = 1$  to  $M$  do
28:     ASSIGNLABELS(assigned $_j$ , child $_{n,j}$ ,  $d + 1$ )
29:   return assigned$ 
```

4.2 Algorithm

In this section we present an algorithm for simultaneously building the classification tree and learning the data representation. We aim at maximizing the accuracy of the tree as defined in Equation 4 by maximizing the objective J_n of Equation 6 at each node of the tree (the boosting theorem that will be presented in Section 5 shows the connection between the two).

Let us now show how we can efficiently optimize the J_n . The gradient of J_n with respect to the conditional probability distributions is (see proof of Lemma 1 in the supplement):

$$\frac{\partial J_n}{\partial p_{j|i}^{(n)}} = \frac{2}{M} q_i^{(n)} (1 - q_i^{(n)}) \text{sign}(p_{j|i}^{(n)} - p_j^{(n)}) \quad (10)$$

Then, according to Equation 10, increasing the likelihood of sending label i to any child j of n such that $p_{j|i}^{(n)} > p_j^{(n)}$ increases the objective J_n . Note that we only need to consider the labels i for which $q_i^{(n)} > 0$, that is, labels i which reach node n in the current tree.

We also want to make sure that each leaf of the tree is only assigned one label. Since a node at depth d has at most M^{D-d+1} descendant leaves, we want to ensure that for a given child j of node n we only increase the value of $p_{j|i}^{(n)}$ for M^{D-d} labels. Algorithm 2 provides such an assignment by greedily choosing the label-child pair (i, j) such that j still has room for labels with the highest value of $\frac{\partial J_n}{\partial p_{j|i}^{(n)}}$.

The global procedure, described in Algorithm 1, is then the following.

- At the start of each batch, re-assign targets for each node prediction function, starting from the root and going down the tree. At each node, each label is more likely to be re-assigned to the child it has had most affinity with in the past (Algorithm 2). This can be seen as a form of hierarchical on-line clustering.
- Every example now has a unique path depending on its label. For each sample, we then take a gradient step at each node along the assigned path (Algorithm 1, lines 22 and 24).

Lemma 1. *Algorithm 2 finds the assignment of nodes to children for a fixed depth tree which most increases J_n . Additionally, for any node n of depth d , if n is currently assigned fewer than $\frac{M^{D-d+1}}{2}$ labels, then Algorithm 1 performs a gradient ascent on J_n .*

Remark 2. *Lemma 1 tells us that as long as the tree is deep enough (i.e. $\frac{M^{D-d+1}}{2}$ is greater than the number of labels that reach any node of depth d), the gradient steps in Algorithm 1 are guaranteed to increase the value of J_n . However, it is sometimes practical to have trees of small depth (e.g. for ease of parallelization on GPUs). While we can loose the theoretical guarantee of Lemma 1 if the tree is too shallow, notice that in that case the gradient step will act to make all $p_{j|i}^{(n)} > p_j^{(n)}$ for the assigned children (i.e. children receiving examples of class i), after which the objective starts increasing again, according to Equation 10.*

Remark 3. *Another interesting feature of the algorithm, is that since the representation of examples from different classes are learned together, there is intuitively less of a risk of getting stuck in a specific tree configuration. More specifically, if two similar classes are initially assigned to different children of a node, the algorithm is less likely to keep this initial decision since the representations for examples of both classes will be pulled together in other nodes.*

Next, we provide a theoretical analysis of the objective introduced in Equation 6. Proofs are deferred to the Supplementary material.

5 Theoretical Results

In this section, we first analyze theoretical properties of the objective J_n as regards node quality, then prove a boosting statement for the global tree accuracy.

5.1 Properties of the objective function

We start by showing that maximizing J_n in every node of the tree leads to high-quality nodes, i.e. perfectly balanced and perfectly pure node splits. Let us first introduce some formal definitions.

Definition 1 (Balancedness factor). *The split in node n of the tree is $\beta^{(n)}$ -balanced if*

$$\beta^{(n)} \leq \min_{j=\{1,2,\dots,M\}} p_j^{(n)},$$

where $\beta^{(n)} \in (0, \frac{1}{M}]$ is a balancedness factor.

A split is perfectly balanced if and only if $\beta^{(n)} = \frac{1}{M}$.

Definition 2 (Purity factor). *The split in node n of the tree is $\alpha^{(n)}$ -pure if*

$$\frac{1}{M} \sum_{j=1}^M \sum_{i=1}^K q_i^{(n)} \min(p_{j|i}^{(n)}, 1 - p_{j|i}^{(n)}) \leq \alpha^{(n)},$$

where $\alpha^{(n)} \in [0, \frac{1}{M})$ is a purity factor.

A split is perfectly pure if and only if $\alpha^{(n)} = 0$.

The following lemmas characterize the range of the objective J_n and link it to the notions of balancedness and purity of the split.

Lemma 2. *The objective function J_n lies in the interval $[0, \frac{4}{M} (1 - \frac{1}{M})]$.*

Let J^* denotes the highest possible value of J_n , i.e. $J^* = \frac{4}{M} (1 - \frac{1}{M})$.

Lemma 3. *The objective function J_n admits the highest value, i.e. $J_n = J^*$, if and only if the split in node n is perfectly balanced, i.e. $\beta^{(n)} = \frac{1}{M}$, and perfectly pure, i.e. $\alpha^{(n)} = 0$.*

We next show Lemmas 4 and 5 which analyze balancedness and purity of a node split in isolation, i.e. we analyze resp. balancedness and purity of a node split when resp. purity and balancedness is fixed and perfect. We show that in such isolated setting increasing J_n leads to a more balanced and more pure split.

Lemma 4. *If a split in node n is perfectly pure, then*

$$\beta^{(n)} \in \left[\frac{1}{M} - \frac{\sqrt{M(J^* - J_n)}}{2}, \frac{1}{M} \right].$$

Lemma 5. *If a split in node n is perfectly balanced, then $\alpha^{(n)} \leq (J^* - J_n)/2$.*

Next we provide a bound on the classification error for the tree. In particular, we show that if the objective is “weakly” optimized in each node of the tree, where this weak advantage is captured in a form of the *Weak Hypothesis Assumption*, then our algorithm will amplify this weak advantage to build a tree achieving any desired level of accuracy.

5.2 Error bound

Denote $y(x)$ to be a fixed target function with domain \mathcal{X} , which assigns the data point x to its label, and let \mathcal{P} be a fixed target distribution over \mathcal{X} . Together y and \mathcal{P} induce a distribution on labeled pairs $(x, y(x))$. Let $t(x)$ be the label assigned to data point x by the tree. We denote as $\epsilon(\mathcal{T})$ the error of tree \mathcal{T} , i.e. $\epsilon(\mathcal{T}) := \mathbb{E}_{x \sim \mathcal{P}} \left[\sum_{i=1}^K \mathbb{1}[t(x) = i, y(x) \neq i] \right]$ ($1 - \epsilon(\mathcal{T})$ refers to the accuracy as given by Equation 4). Then the following theorem holds

Theorem 1. *The Weak Hypothesis Assumption says that for any distribution \mathcal{P} over the data, at each node n of the tree \mathcal{T} there exists a partition such that $J_n \geq \gamma$, where $\gamma \in \left[\frac{M}{2} \min_{j=1,2,\dots,M} p_j, 1 - \frac{M}{2} \min_{j=1,2,\dots,M} p_j \right]$.*

Under the Weak Hypothesis Assumption, for any $\kappa \in [0, 1]$, to obtain $\epsilon(\mathcal{T}) \leq \kappa$ it suffices to have a tree with

$$N \geq \left(\frac{1}{\kappa} \right)^{\frac{16[M(1-2\gamma)+2\gamma](M-1)}{\log_2 e M^2 \gamma^2} \ln K} \text{ internal nodes.}$$

The above theorem shows the number of splits that suffice to reduce the multi-class classification error of the tree below an arbitrary threshold κ . As shown in the proof of the above theorem in the Supplement, the *Weak Hypothesis Assumption* implies that all p_j s satisfy: $p_j \in [\frac{2\gamma}{M}, \frac{M(1-2\gamma)+2\gamma}{M}]$. Below we show a tighter version of this bound when assuming that each node induces balanced split.

Corollary 1. *The Weak Hypothesis Assumption says that for any distribution \mathcal{P} over the data, at each node n of the tree \mathcal{T} there exists a partition such that $J_n \geq \gamma$, where $\gamma \in \mathbb{R}^+$.*

Under the Weak Hypothesis Assumption and when all nodes make perfectly balanced splits, for any $\kappa \in [0, 1]$, to obtain $\epsilon(\mathcal{T}) \leq \kappa$ it suffices to have a tree with

$$N \geq \left(\frac{1}{\kappa} \right)^{\frac{16(M-1)}{\log_2 e M^2 \gamma^2} \ln K} \text{ internal nodes.}$$

6 Application to Language Modeling

We now show how to adapt the algorithm presented in Section 4 for conditional density estimation, more

specifically how to apply it to language modeling.

Hierarchical Log Bi-Linear Language Model (HLBL) We take the same approach to language modeling as [MH09]. First, using the chain rule and an order T Markov assumption we model the probability of a sentence $\mathbf{w} = (w_1, w_2, \dots, w_n)$ as:

$$p(w_1, w_2, \dots, w_n) = \prod_{t=1}^n p(w_t | w_{t-T}, \dots, w_{t-1})$$

Similarly to their work, we also use a low dimensional representation of the context $(w_{t-T}, \dots, w_{t-1})$. In this setting, each word w in the vocabulary \mathcal{V} has an embedding $U_w \in \mathbb{R}^{d_r}$. A given context $x = (w_{t-T}, \dots, w_{t-1})$ corresponding to position t is then represented by a context embedding vector r_x such that

$$r_x = \sum_{k=1}^T R_k U_{w_{t-k}},$$

where $U \in \mathbb{R}^{|\mathcal{V}| \times d_r}$ is the embedding matrix, and $R_k \in \mathbb{R}^{d_r \times d_r}$ is the transition matrix associated with the k^{th} context word.

The most straight-forward way to define a probability function is then to define the distribution over the next word given the context representation as a soft-max, as done in [MH07]. That is:

$$\begin{aligned} p(w_t = i | x) &= \sigma_i(r_x^\top U + \mathbf{b}) \\ &= \frac{\exp(r_x^\top U_i + b_i)}{\sum_{w \in \mathcal{V}} \exp(r_x^\top U_w + b_w)}, \end{aligned}$$

where b_w is the bias for word w . However, the complexity of computing this probability distribution in this setting is $O(|\mathcal{V}| \times d_r)$, which can be prohibitive for large corpora and vocabularies.

Instead, [MH09] takes a hierarchical approach to the problem. They construct a binary tree, where each word $w \in \mathcal{V}$ corresponds to some leaf of the tree, and can thus be identified with the path from the root to the corresponding leaf by making a sequence of choices of going left versus right. This corresponds to the tree-structured log-likelihood objective presented in Equation 5 for the case where $M = 2$, and $f_\Theta(x) = r_x$. More precisely, if \mathbf{c}^i is the path to word i as defined in Expression 3, then:

$$\log p(w_t = i | x) = \sum_{d=1}^D \log \sigma_{c_{d,2}}((r_x^\top U^{c_{d,1}} + \mathbf{b}^{c_{d,1}})) \quad (11)$$

In this binary case, σ is the sigmoid function, and for all non-leaf nodes $n \in \{1, 2, \dots, N\}$, we have $U^n \in \mathbb{R}^{d_r}$ and $\mathbf{b}^n \in \mathbb{R}^{d_r}$. The cost of computing the likelihood of word w is then reduced to $O(\log(|\mathcal{V}|) \times d_r)$. In their work, the authors start the training procedure by using a random tree, then alternate parameter learning with using a clustering-based heuristic to rebuild their hierarchy. We expand upon their method by providing an algorithm which allows for using hierarchies of arbitrary width, and jointly learns the tree structure and the model parameters.

Using our Algorithm We may use Algorithm 1 as is to learn a good tree structure for classification: that is, a model that often predicts w_t to be the most likely word after seeing the context $(w_{t-T}, \dots, w_{t-1})$. However, while this could certainly learn interesting representations and tree structure, there is no guarantee that such a model would achieve a good average log-likelihood. Intuitively, there are often several valid possibilities for a word given its immediate left context, which a classification objective does not necessarily take into account. Yet another option would be to learn a tree structure that maximizes the classification objective, then fine-tune the model parameters using the log-likelihood objective. We tried this method, but initial tests of this approach did not do much better than the use of random trees. Instead, we present here a small modification of Algorithm 1 which is equivalent to log-likelihood training when restricted to the fixed tree setting, and can be shown to increase the value of the node objectives J_n . In lines 11 and 12 of Algorithm 1, as well as in line 21 of Algorithm 2, we can re-place the gradients with respect to p_{target} by these with respect to $\log p_{\text{target}}$. Then, for a given tree structure, the algorithm takes a gradient step with respect to the log-likelihood of the samples:

$$\frac{\partial J_n}{\partial \log p_{j|i}^{(n)}} = \frac{2}{M} q_i^{(n)} (1 - q_i^{(n)}) \text{sign}(p_{j|i}^{(n)} - p_j^{(n)}) p_{j|i}^{(n)}. \quad (12)$$

Lemma 1 extends to the new version of the algorithm.

7 Experiments

We ran language modeling experiments on two text data sets: Penn TreeBank (PTB), which consists of about 1M tokens, with a vocabulary of 10,000 words, and the Gutenberg novel corpus, which has about 50M tokens and a vocabulary of 250,000 words. We used a 12GB NVIDIA GeForce GTX TITAN GPU. All experiments use trees of depth 3 (25-ary for PTB, 75-ary for Gutenberg).

Model	PTB		Gutenberg	
	perp.	ms/batch	perp.	ms/batch
Random Tree	172	61	160	111
Learned Tree	159	63	148	112
Flat soft-max	151	105	149	198

Table 1: Comparison of a flat soft-max to a depth 3 hierarchical soft-max (learned and random tree).

Table 1 presents perplexity results on two language modeling tasks. On PTB, we don't quite reach the perplexity of the non-hierarchical (flat) soft-max, but there is a definite advantage to learning the tree over using a random structure (for which we provide the best of 5 initializations). On Gutenberg, the learned tree model is twice as fast as the flat objective without losing any points of perplexity. We compare the running time of our implementation of depth 3 trees to the

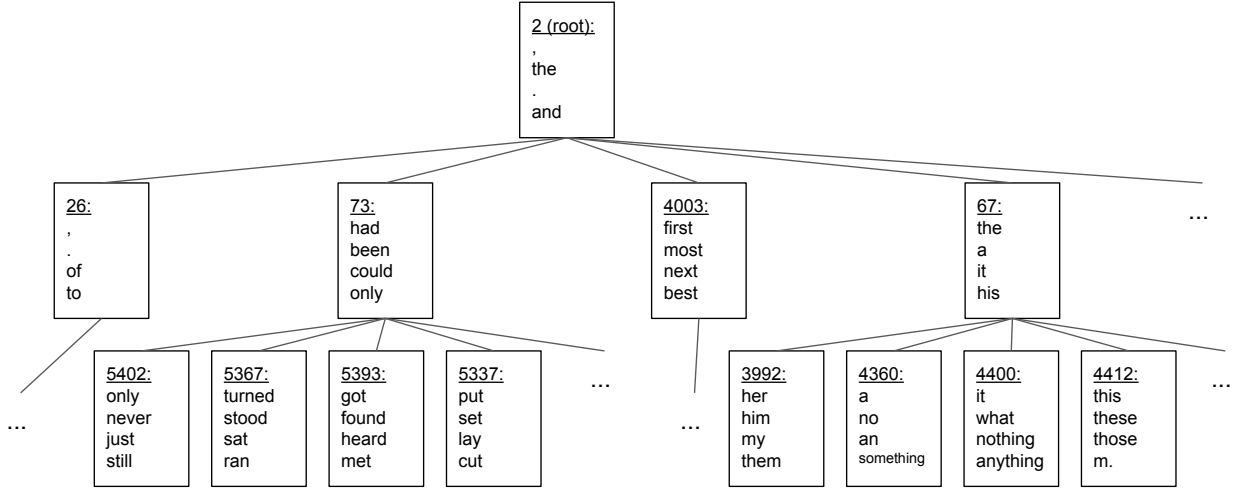


Figure 4: Tree learned from the Gutenberg corpus, showing the four most common words assigned to each node.

standard Torch implementation of the flat soft-max. While we manage to obtain a significant speed-up, it is our belief that a specific purpose GPU implementation would lead to even greater acceleration and empirically realize the advantage of having a $O(\log_M(|\mathcal{V}|) \times M \times d_r)$ instead of $O(|\mathcal{V}| \times d_r)$ computational complexity. We use Algorithm 2 to re-build the tree twenty times per epoch, thus the effect of it on the run time is negligible.

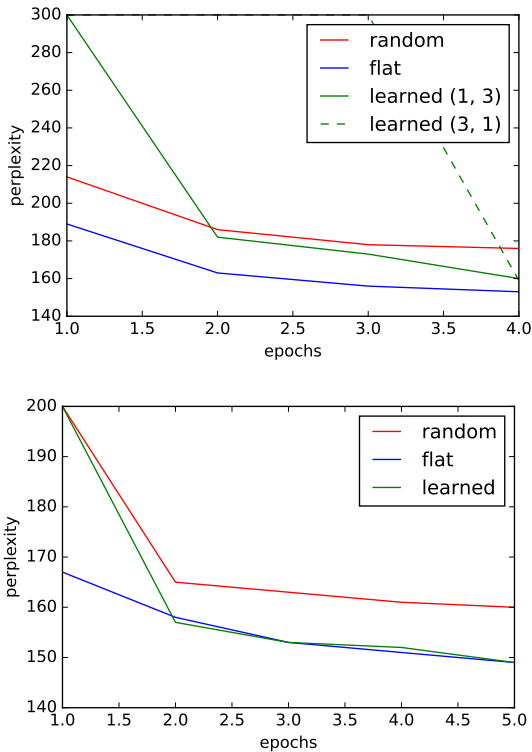


Figure 3: Test perplexity per epoch. Top: Penn Tree Bank. Bottom: Gutenberg.

Figure 3 shows the evolution of the test perplexity for a few epochs. We explored two settings on PTB: either learn the tree for 1 epoch and fine tune the parameters for 3, or learn the tree for 3 epochs and fine tune

for 1. We compare to a random tree and flat soft-max trained for 4 epochs. While the second setting performs slightly better, it appears that most of the relevant tree structure can be learned in one epoch. On Gutenberg, after one epoch of learning the tree with our algorithm, the hierarchical soft-max performs similarly to the flat one.

Leaf 229	Leaf 230	Leaf 300	Leaf 231
suggested	vegas	payments	operates
watched	&	buy-outs	includes
created	calif.	swings	intends
violated	park	gains	makes
introduced	n.j.	taxes	means
discovered	conn.	operations	helps
carried	pa.	profits	seeks

Table 2: Example of leaf nodes for the tree learned on PTB. We can identify a leaf for 3rd person verbs, one for past participles, one for plural nouns, and one (loosely) for places.

Figure 4 shows a sub-set of the tree learned on the Gutenberg dataset, while Table 2 presents some leaves of the PTB learned tree. Both show that the algorithm learns a coherent tree structure.

8 Conclusion

In this paper, we introduced a provably accurate algorithm for jointly learning tree structure and parameters for hierarchical prediction. We applied it to the problem of language modeling, and showed a substantial speed up over a non-hierarchical approach without sacrificing performance.

This work gives rise to a number of interesting further questions, including looking for ways to take advantage of the tree building method to handle polysemous labels and studying the convergence properties of our algorithm (empirically observed).

References

- [AGNS11] A. Azocar, J. Gimenez, K. Nikodem, and J. L. Sanchez. On strongly midconvex functions. *Opuscula Math.*, 31(1):15–26, 2011.
- [AGPV13] R. Agarwal, A. Gupta, Y. Prabhu, and M. Varma. Multi-label learning with millions of labels: Recommending advertiser bid phrases for web pages. In *WWW*, 2013.
- [AK15] J. Andreas and D. Klein. When and why are log-linear models self-normalizing? In *NAACL HLT*, 2015.
- [AKK⁺14] A. Agarwal, S. M. Kakade, N. Karampatziakis, L. Song, and G. Valiant. Least squares revisited: Scalable approaches for multi-class prediction. In *ICML*, 2014.
- [BdM⁺92] P. F. Brown, P. V. deSouza, R. L. Mercer, V. J. Della Pietra, and J. C. Lai. Class-based n-gram models of natural language. *Comput. Linguist.*, 18(4):467–479, 1992.
- [BDVJ03a] Y. Bengio, R. Ducharme, Pascal V., and C. Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, 2003.
- [BDVJ03b] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.
- [BFOS84] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. CRC Press LLC, Boca Raton, Florida, 1984.
- [Bis06] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
- [BJK⁺15] K. Bhatia, H. Jain, P. Kar, M. Varma, and P. Jain. Sparse local embeddings for extreme multi-label classification. In *NIPS*. 2015.
- [BLL⁺09] A. Beygelzimer, J. Langford, Y. Lifshits, G. B. Sorkin, and A. L. Strehl. Conditional probability tree estimation analysis and algorithms. In *UAI*, 2009.
- [BLR09] A. Beygelzimer, J. Langford, and P. D. Ravikumar. Error-correcting tournaments. In *ALT*, 2009.
- [BM98] L. Douglas Baker and Andrew Kachites McCallum. Distributional clustering of words for text classification. In *SIGIR*, 1998.
- [Bot98] L. Bottou. Online algorithms and stochastic approximations. In *Online Learning and Neural Networks*. Cambridge University Press, 1998.
- [Bre01] L. Breiman. Random forests. *Mach. Learn.*, 45:5–32, 2001.
- [BS03] Y. Bengio and J.-S. S  n  cal. Quick training of probabilistic neural nets by importance sampling. In *AISTATS*, 2003.
- [BS08] Y. Bengio and J.-S. Sen  cal. Adaptive importance sampling to accelerate training of a neural probabilistic language model. *IEEE Trans. Neural Networks*, 19:713–722, 2008.
- [BSKV14] O. Beijbom, M. Saberian, D. Kriegman, and N. Vasconcelos. Guess-averse loss functions for cost-sensitive multiclass boosting. In *ICML*, 2014.
- [BWG10] S. Bengio, J. Weston, and D. Grangier. Label embedding trees for large multi-class tasks. In *NIPS*, 2010.
- [CCB16] A. Choromanska, K. Choromanski, and M. Bojarski. On the boosting ability of top-down decision tree learning algorithm for multiclass classification. *CoRR*, abs/1605.05223, 2016.
- [CG96] S. F. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. In *ACL*, 1996.
- [CKF11] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, number EPFL-CONF-192376, 2011.
- [CL15] A. Choromanska and J. Langford. Logarithmic time online multiclass prediction. In *NIPS*. 2015.
- [CW08] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*, 2008.
- [dBV16] A. de Br  bisson and P. Vincent. An exploration of softmax alternatives belonging to the spherical loss family. In *ICLR*, 2016.
- [DKLM16] H. Daume, N. Karampatziakis, J. Langford, and P. Mineiro. Logarithmic time one-against-some. *CoRR*, abs/1606.04988, 2016.
- [DSBFF11] J. Deng, S. Satheesh, A. C. Berg, and L. Fei-Fei. Fast and balanced: Efficient label tree learning for large scale object recognition. In *NIPS*, 2011.
- [DWR⁺15] N. Djuric, H. Wu, V. Radosavljevic, M. Grbovic, and N. Bhamidipati. Hierarchical neural language models for joint representation of streaming documents and their content. In *WWW*, 2015.
- [GH12] M. U. Gutmann and A. Hyv  rinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *J. Mach. Learn. Res.*, 13(1):307–361, 2012.
- [GJC⁺16] E. Grave, A. Joulin, M. Ciss  , D. Grangier, and H. J  gou. Efficient softmax approximation for gpus. *CoRR*, abs/1609.04309, 2016.
- [GV13] J. Gubbins and A. Vlachos. Dependency language models for sentence completion. In *EMNLP*, 2013.

-
- [HKLZ09] D. Hsu, S. Kakade, J. Langford, and T. Zhang. Multi-label prediction via compressed sensing. In *NIPS*, 2009.
- [JM80] F. Jelinek and R. L. Mercer. Interpolated estimation of Markov source parameters from sparse data. In *Proceedings, Workshop on Pattern Recognition in Practice*, pages 381–397. North Holland, 1980.
- [Kat87] S. M. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. In *IEEE Trans. on Acoustics, Speech and Singal proc.*, volume ASSP-35, pages 400–401, 1987.
- [KFCB15] P. Kotschieder, M. Fiterau, A. Criminisi, and S. Rota Bulò. Deep Neural Decision Forests. In *ICCV*, 2015.
- [KIS⁺15] A. Kumar, O. Irsoy, J. Su, J. Bradbury, R. English, B. Pierce, P. Ondruska, I. Gulrajani, and R. Socher. Ask me anything: Dynamic memory networks for natural language processing. *CoRR*, abs/1506.07285, 2015.
- [KN93] R. Kneser and H. Ney. Improved clustering techniques for class-based statistical language modelling. In *EUROSPEECH*, 1993.
- [LLS07] J. Langford, L. Li, and A. Strehl. <http://hunch.net/~vw>, 2007.
- [MB05] F. Morin and Y. Bengio. Hierarchical probabilistic neural network language model. In *AISTATS*, 2005.
- [MDK⁺11] T. Mikolov, A. Deoras, S. Kombrink, L. Burget, and J. Honza Cernocky. Empirical evaluation and combination of advanced language modeling techniques. In *INTERSPEECH*, 2011.
- [MGC09] G. Madzarov, D. Gjorgjevikj, and I. Chorbev. A multi-class svm classifier utilizing binary decision tree. *Informatica*, 33(2):225–233, 2009.
- [MH07] A. Mnih and G. Hinton. Three new graphical models for statistical language modelling. In *ICML*, 2007.
- [MH09] A. Mnih and G. E. Hinton. A scalable hierarchical distributed language model. In *NIPS*. 2009.
- [MKB⁺10] T. Mikolov, M. Karafit, L. Burget, J. Cernock, and S. Khudanpur. Recurrent neural network based language model. In *INTERSPEECH*, 2010.
- [MN10] A. L. Maas and A. Y. Ng. A probabilistic model for semantic word vectors. In *Workshop on Deep Learning and Unsupervised Feature Learning, NIPS*, 2010.
- [MSC⁺13] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013.
- [MT12] A. Mnih and Y. W. Teh. A fast and simple algorithm for training neural probabilistic language models. In *ICML*, 2012.
- [MV15] P. Mirowski and A. Vlachos. Dependency recurrent neural language models for sentence completion. *CoRR*, abs/1507.01193, 2015.
- [NWW98] T.R. Niesler, E. W. D. Whittaker, and P.C. Woodland. Comparison of part-of-speech and automatically derived category-based language models for speech recognition. In *ICASSP*, 1998.
- [PTL93] F. Pereira, N. Tishby, and L. Lee. Distributional clustering of English words. In *ACL*, 1993.
- [PV14] Y. Prabhu and M. Varma. Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *ACM SIGKDD*, 2014.
- [RG11] P. Ram and A. G. Gray. Density estimation trees. In *KDD*, 2011.
- [Sch04] H. Schwenk. Efficient training of large neural networks for language modeling. In *IJCNN*, 2004.
- [SG02] H. Schwenk and J.-L. Gauvain. Connectionist language modeling for large vocabulary continuous speech recognition. In *ICASSP*, 2002.
- [SG05] H. Schwenk and J.-L. Gauvain. Training neural network language models on very large corpora. In *HLT*, 2005.
- [SLNM11] R. Socher, C. C.-Y. Lin, A. Y. Ng, and C. D. Manning. Parsing natural scenes and natural language with recursive neural networks. In *ICML*, 2011.
- [SS12] S. Shalev-Shwartz. Online learning and online convex optimization. *Found. Trends Mach. Learn.*, 4(2):107–194, 2012.
- [TRB10] J. Turian, L. Ratinov, and Y. Bengio. Word representations: A simple and general method for semi-supervised learning. In *ACL*, 2010.
- [TSM15] K. S. Tai, R. Socher, and C. D. Manning. Improved semantic representations from tree-structured long short-term memory networks. *CoRR*, abs/1503.00075, 2015.
- [VdBB15] P. Vincent, A. de Brébisson, and X. Bouthillier. Efficient exact gradient update for training deep networks with very large sparse targets. In *NIPS*, 2015.
- [WBU11] J. Weston, S. Bengio, and N. Usunier. Wsabie: Scaling up to large vocabulary image annotation. In *IJCAI*, 2011.
- [WMY13] J. Weston, A. Makadia, and H. Yee. Label partitioning for sublinear ranking. In *ICML*, 2013.
- [XEJ03] P. Xu, A. Emami, and F. Jelinek. Training connectionist models for the structured language model. In *EMNLP*, 2003.
- [ZPM⁺12] G. Zweig, J. C. Platt, C. Meek, C. J.C. Burges, A. Yessenalina, and Q. Liu. Computational approaches to sentence completion. In *ACL*, 2012.

- [ZX13] B. Zhao and E. P. Xing. Sparse output coding for large-scale visual recognition. In *CVPR*, 2013.

Simultaneous Learning of Trees and Representations for Extreme Classification with Application to Language Modeling (Supplementary material)

9 Theoretical proofs

Proof of Lemma 1. Recall the form of the objective defined in 6:

$$\begin{aligned} J_n &= \frac{2}{M} \sum_{i=1}^K q_i^{(n)} \left(\sum_{j=1}^M |p_j^{(n)} - p_{j|i}^{(n)}| \right) \\ &= \frac{2}{M} \mathbb{E}_{i \sim q^{(n)}} \left[f_n^J(i, p_{\cdot|\cdot}^{(n)}, q^{(n)}) \right] \end{aligned}$$

Where:

$$\begin{aligned} f_n^J(i, p_{\cdot|\cdot}^{(n)}, q^{(n)}) &= \sum_{j=1}^M \left| p_j^{(n)} - p_{j|i}^{(n)} \right| \\ &= \sum_{j=1}^M \left| p_{j|i}^{(n)} - \sum_{i'=1}^K q_{i'}^{(n)} p_{j|i'}^{(n)} \right| \\ &= \sum_{j=1}^M \left| \sum_{i'=1}^K (\mathbb{1}_{i=i'} - q_{i'}^{(n)}) p_{j|i'}^{(n)} \right| \end{aligned}$$

Hence:

$$\frac{\partial f_n^J(i, p_{\cdot|\cdot}^{(n)}, q^{(n)})}{\partial p_{j|i}^{(n)}} = (1 - q_i^{(n)}) \text{sign}(p_{j|i}^{(n)} - p_j^{(n)})$$

And:

$$\begin{aligned} \frac{\partial f_n^J(i, p_{\cdot|\cdot}^{(n)}, q^{(n)})}{\partial \log p_{j|i}^{(n)}} &= (1 - q_i^{(n)}) \text{sign}(p_{j|i}^{(n)} - p_j^{(n)}) \frac{\partial p_{j|i}^{(n)}}{\partial \log p_{j|i}^{(n)}} \\ &= (1 - q_i^{(n)}) \text{sign}(p_{j|i}^{(n)} - p_j^{(n)}) p_{j|i}^{(n)} \end{aligned}$$

By assigning each label j to a specific child i under the constraint that no child has more than L labels, we take a step in the direction $\partial E \in \{0, 1\}^{M \times K}$, where:

$$\begin{aligned} \forall i \in [1, K], \quad \sum_{j=1}^M \partial E_{j,i} &= 1 \\ \text{and} \\ \forall j \in [1, M], \quad \sum_{i=1}^K \partial E_{j,i} &\leq L \end{aligned}$$

Thus:

$$\begin{aligned} \frac{\partial J_n}{\partial p_{\cdot|\cdot}^{(n)}} \partial E &= \frac{2}{M} \frac{\mathbb{E}_{i \sim q^{(n)}} \left[f_n^J(i, p_{\cdot|\cdot}^{(n)}, q^{(n)}) \right]}{\partial p_{\cdot|\cdot}^{(n)}} \partial E \\ &= \frac{2}{M} \sum_{i=1}^K q_i^{(n)} (1 - q_i^{(n)}) \sum_{j=1}^M \left(\text{sign}(p_{j|i}^{(n)} - p_j^{(n)}) \partial E_{j,i} \right) \end{aligned} \tag{13}$$

And:

$$\frac{\partial J_n}{\partial \log p_{\cdot|\cdot}} \partial E = \frac{2}{M} \sum_{i=1}^K q_i^{(n)} (1 - q_i^{(n)}) \sum_{j=1}^M \left(\text{sign}(p_{j|i}^{(n)} - p_j^{(n)}) p_{j|i}^{(n)} \partial E_{j,i} \right) \quad (14)$$

If there exists such an assignment for which 13 is positive, then the greedy method proposed in 2 finds it. Indeed, suppose that Algorithm 2 assigns label i to child j and i' to j' . Suppose now that another assignment $\partial E'$ sends i to j' and i' to j . Then:

$$\frac{\partial J_n}{\partial p_{\cdot|\cdot}} (\partial E - \partial E') = \left(\frac{\partial J_n}{\partial p_{j|i}^{(n)}} + \frac{\partial J_n}{\partial p_{j'|i'}^{(n)}} \right) - \left(\frac{\partial J_n}{\partial p_{j|i'}^{(n)}} + \frac{\partial J_n}{\partial p_{j'|i}^{(n)}} \right) \quad (15)$$

Since the algorithm assigns children by descending order of $\frac{\partial J_n}{\partial p_{j|i}^{(n)}}$ until a child j is full, we have:

$$\frac{\partial J_n}{\partial p_{j|i}^{(n)}} \geq \frac{\partial J_n}{\partial p_{j|i'}^{(n)}} \quad \text{and} \quad \frac{\partial J_n}{\partial p_{j'|i'}^{(n)}} \geq \frac{\partial J_n}{\partial p_{j'|i}^{(n)}}$$

Hence:

$$\frac{\partial J_n}{\partial p_{\cdot|\cdot}} (\partial E - \partial E') \geq 0$$

Thus, the greedy algorithm finds the assignment that most increases J_n most under the children size constraints.

Now, consider the case where $L \geq \frac{K}{2}$. This means that at least $\frac{K}{2}$ labels are assigned to a child such that $p_{j|i}^{(n)} \geq p_j^{(n)}$.

In the standard version of the algorithm, the absolute value of $\frac{\partial J_n}{\partial p_{j|i}^{(n)}}$ is the same for all values of j , hence the labels with positive contributions to the gradient are the one with the highest value, and:

$$\frac{\partial J_n}{\partial p_{\cdot|\cdot}} \partial E \geq 0 \quad (16)$$

□

Proof of Lemma 2. Both J_n and J_T are defined as the sum of non-negative values which gives the lower-bound. We next derive the upper-bound on J_n . Recall:

$$J_n = \frac{2}{M} \sum_{j=1}^M \sum_{i=1}^K q_i^{(n)} |p_j^{(n)} - p_{j|i}^{(n)}| = \frac{2}{M} \sum_{j=1}^M \sum_{i=1}^K q_i^{(n)} \left| \sum_{l=1}^K q_l^{(n)} p_{j|l}^{(n)} - p_{j|i}^{(n)} \right|$$

since $p_j^{(n)} = \sum_{l=1}^K q_l^{(n)} p_{j|l}^{(n)}$. The objective J_n is maximized on the extremes of the $[0, 1]$ interval. Thus, define the following two sets of indices:

$$O_j = \{i : i \in \{1, 2, \dots, K\}, p_{j|i}^{(n)} = 1\} \quad \text{and} \quad Z_j = \{i : i \in \{1, 2, \dots, K\}, p_{j|i}^{(n)} = 0\}.$$

We omit indexing these sets with n for the ease of notation. We continue as follows

$$\begin{aligned} J_n &\leq \frac{2}{M} \sum_{j=1}^M \left[\sum_{i \in O_j} q_i^{(n)} \left(1 - \sum_{l \in O_j} q_l^{(n)} \right) + \sum_{i \in Z_j} q_i^{(n)} \sum_{l \in O_j} q_l^{(n)} \right] \\ &= \frac{4}{M} \sum_{j=1}^M \left[\sum_{i \in O_j} q_i^{(n)} - \left(\sum_{i \in O_j} q_i^{(n)} \right)^2 \right] \\ &= \frac{4}{M} \left[1 - \sum_{j=1}^M \left(\sum_{i \in O_j} q_i^{(n)} \right)^2 \right], \end{aligned}$$

where the last inequality is the consequence of the following: $\sum_{j=1}^M p_j^{(n)} = 1$ and $p_j^{(n)} = \sum_{l=1}^K q_l^{(n)} p_{j|l}^{(n)} = \sum_{i \in O_j} q_i^{(n)}$, thus $\sum_{j=1}^M \sum_{i \in O_j} q_i^{(n)} = 1$. Applying Jensen's inequality to the last inequality obtained gives

$$\begin{aligned} J_n &\leq \frac{4}{M} - 4 \left[\sum_{j=1}^M \left(\frac{1}{M} \sum_{i \in O_j} q_i^{(n)} \right) \right]^2 \\ &= \frac{4}{M} \left(1 - \frac{1}{M} \right) \end{aligned}$$

That ends the proof. \square

Proof of Lemma 3. We start from proving that if the split in node n is perfectly balanced, i.e. $\forall_{j=\{1,2,\dots,M\}} p_j^{(n)} = \frac{1}{M}$, and perfectly pure, i.e. $\forall_{j=\{1,2,\dots,M\}} \min_{i=\{1,2,\dots,K\}} (p_{j|i}^{(n)}, 1 - p_{j|i}^{(n)}) = 0$, then J_n admits the highest value $J_n = \frac{4}{M} (1 - \frac{1}{M})$. Since the split is maximally balanced we write:

$$J_n = \frac{2}{M} \sum_{j=1}^M \sum_{i=1}^K q_i^{(n)} \left| \frac{1}{M} - p_{j|i}^{(n)} \right|.$$

Since the split is maximally pure, each $p_{j|i}^{(n)}$ can only take value 0 or 1. As in the proof of previous lemma, define two sets of indices:

$$O_j = \{i : i \in \{1, 2, \dots, K\}, p_{j|i}^{(n)} = 1\} \quad \text{and} \quad Z_j = \{i : i \in \{1, 2, \dots, K\}, p_{j|i}^{(n)} = 0\}.$$

We omit indexing these sets with n for the ease of notation. Thus

$$\begin{aligned} J_n &= \frac{2}{M} \sum_{j=1}^M \left[\sum_{i \in O_j} q_i^{(n)} \left(1 - \frac{1}{M} \right) + \sum_{i \in Z_j} q_i^{(n)} \frac{1}{M} \right] \\ &= \frac{2}{M} \sum_{j=1}^M \left[\sum_{i \in O_j} q_i^{(n)} \left(1 - \frac{1}{M} \right) + \frac{1}{M} \left(1 - \sum_{i \in O_j} q_i^{(n)} \right) \right] \\ &= \frac{2}{M} \left(1 - \frac{2}{M} \right) \sum_{j=1}^M \sum_{i \in O_j} q_i^{(n)} + \frac{2}{M} \\ &= \frac{4}{M} \left(1 - \frac{1}{M} \right), \end{aligned}$$

where the last equality comes from the fact that $\sum_{j=1}^M p_j^{(n)} = 1$ and $p_j^{(n)} = \sum_{l=1}^K q_l^{(n)} p_{j|l}^{(n)} = \sum_{i \in O_j} q_i^{(n)}$, thus $\sum_{j=1}^M \sum_{i \in O_j} q_i^{(n)} = 1$.

Thus we are done with proving one induction direction. Next we prove that if J_n admits the highest value $J_n = \frac{4}{M} (1 - \frac{1}{M})$, then the split in node n is perfectly balanced, i.e. $\forall_{j=\{1,2,\dots,M\}} p_j^{(n)} = \frac{1}{M}$, and perfectly pure, i.e. $\forall_{j=\{1,2,\dots,M\}} \min_{i=\{1,2,\dots,K\}} (p_{j|i}^{(n)}, 1 - p_{j|i}^{(n)}) = 0$.

Without loss of generality assume each $q_i^{(n)} \in (0, 1)$. The objective J_n is certainly maximized in the extremes of the interval $[0, 1]$, where each $p_{j|i}^{(n)}$ is either 0 or 1. Also, at maximum it cannot be that for any given j , all $p_{j|i}^{(n)}$'s are 0 or all $p_{j|i}^{(n)}$'s are 1. The function $J(h)$ is differentiable in these extremes. Next, define three sets of indices:

$$\mathcal{A}_j = \{i : \sum_{l=1}^K q_l^{(n)} p_{j|l}^{(n)} \geq p_{j|i}^{(n)}\} \quad \text{and} \quad \mathcal{B}_j = \{i : \sum_{l=1}^K q_l^{(n)} p_{j|l}^{(n)} < p_{j|i}^{(n)}\} \quad \text{and} \quad \mathcal{C}_j = \{i : \sum_{l=1}^K q_l^{(n)} p_{j|l}^{(n)} > p_{j|i}^{(n)}\}.$$

We omit indexing these sets with n for the ease of notation. Objective J_n can then be re-written as

$$J_n = \frac{2}{M} \sum_{j=1}^M \left[\sum_{i \in \mathcal{A}_j} q_i^{(n)} \left(\sum_{l=1}^K q_i^{(n)} p_{j|l}^{(n)} - p_{j|i}^{(n)} \right) + 2 \sum_{i \in \mathcal{B}_j} q_i^{(n)} \left(p_{j|i}^{(n)} - \sum_{l=1}^K q_i^{(n)} p_{j|l}^{(n)} \right) \right],$$

We next compute the derivatives of J_n with respect to $p_{j|z}^{(n)}$, where $z = \{1, 2, \dots, K\}$, everywhere where the function is differentiable and obtain

$$\frac{\partial J_n}{\partial p_{j|z}^{(n)}} = \begin{cases} 2q_z^{(n)}(\sum_{i \in \mathcal{C}_j} q_i^{(n)} - 1) & \text{if } z \in \mathcal{C}_j \\ 2q_z^{(n)}(1 - \sum_{i \in \mathcal{B}_j} q_i^{(n)}) & \text{if } z \in \mathcal{B}_j \end{cases},$$

Note that in the extremes of the interval $[0, 1]$ where J_n is maximized, it cannot be that $\sum_{i \in \mathcal{C}_j} q_i^{(n)} = 1$ or $\sum_{i \in \mathcal{B}_j} q_i^{(n)} = 1$ thus the gradient is non-zero. This fact and the fact that J_n is convex imply that J_n can *only* be maximized at the extremes of the $[0, 1]$ interval. Thus if J_n admits the highest value, then the node split is perfectly pure. We still need to show that if J_n admits the highest value, then the node split is also perfectly balanced. We give a proof by contradiction, thus we assume that at least for one value of j , $p_j^{(n)} \neq \frac{1}{M}$, or in other words if we decompose each $p_j^{(n)}$ as $p_j^{(n)} = \frac{1}{M} + x_j$, then at least for one value of j , $x_j \neq 0$. Lets once again define two sets of indices (we omit indexing x_j and these sets with n for the ease of notation):

$$O_j = \{i : i \in \{1, 2, \dots, K\}, p_{j|i}^{(n)} = 1\} \quad \text{and} \quad Z_j = \{i : i \in \{1, 2, \dots, K\}, p_{j|i}^{(n)} = 0\},$$

and recall that $p_j^{(n)} = \sum_{l=1}^K q_l^{(n)} p_{j|l}^{(n)} = \sum_{i \in O_j} q_i^{(n)}$. We proceed as follows

$$\begin{aligned} \frac{4}{M} \left(1 - \frac{1}{M} \right) = J_n &= \frac{2}{M} \sum_{j=1}^M \left[\sum_{i \in O_j} q_i^{(n)} (1 - p_j^{(n)}) + \sum_{i \in Z_j} q_i^{(n)} p_j^{(n)} \right] \\ &= \frac{2}{M} \sum_{j=1}^M \left[p_j^{(n)} (1 - p_j^{(n)}) + p_j^{(n)} (1 - p_j^{(n)}) \right] \\ &= \frac{4}{M} \sum_{j=1}^M \left[p_j^{(n)} - (p_j^{(n)})^2 \right] \\ &= \frac{4}{M} \left[1 - \sum_{j=1}^M (p_j^{(n)})^2 \right] \\ &= \frac{4}{M} \left[1 - \sum_{j=1}^M \left(\frac{1}{M} + x_j \right)^2 \right] \\ &= \frac{4}{M} \left(1 - \frac{1}{M} - \frac{2}{M} \sum_{j=1}^M x_j - \sum_{j=1}^M x_j^2 \right) \\ &< \frac{4}{M} \left(1 - \frac{1}{M} \right) \end{aligned}$$

Thus we obtain the contradiction which ends the proof. \square

Proof of Lemma 4. Since we node that the split is perfectly pure, then each $p_{j|i}^{(n)}$ is either 0 or 1. Thus we define two sets

$$O_j = \{i : i \in \{1, 2, \dots, K\}, p_{j|i}^{(n)} = 1\} \quad \text{and} \quad Z_j = \{i : i \in \{1, 2, \dots, K\}, p_{j|i}^{(n)} = 0\}.$$

and thus

$$J_n = \frac{2}{M} \sum_{j=1}^M \left[\sum_{i \in O_j} q_i^{(n)} (1 - p_j) + \sum_{i \in Z_j} q_i^{(n)} p_j \right]$$

Note that $p_j = \sum_{i \in O_j} q_i^{(n)}$. Then

$$J_n = \frac{2}{M} \sum_{j=1}^M [p_j (1 - p_j) + (1 - p_j) p_j] = \frac{4}{M} \sum_{j=1}^M p_j (1 - p_j) = \frac{4}{M} \left(1 - \sum_{j=1}^M p_j^2 \right)$$

and thus

$$\sum_{j=1}^M p_j^2 = 1 - \frac{MJ_n}{4}. \quad (17)$$

Lets express p_j as $p_j = \frac{1}{M} + \epsilon_j$, where $\epsilon_j \in [-\frac{1}{M}, 1 - \frac{1}{M}]$. Then

$$\sum_{j=1}^M p_j^2 = \sum_{j=1}^M \left(\frac{1}{M} + \epsilon_j \right)^2 = \frac{1}{M} + \frac{2}{M} \sum_{j=1}^M \epsilon_j + \sum_{j=1}^M \epsilon_j^2 = \frac{1}{M} + \sum_{j=1}^M \epsilon_j^2, \quad (18)$$

since $\frac{2}{M} \sum_{j=1}^M \epsilon_j = 0$. Thus combining Equation 17 and 18

$$\frac{1}{M} + \sum_{j=1}^M \epsilon_j^2 = 1 - \frac{MJ_n}{4}$$

and thus

$$\sum_{j=1}^M \epsilon_j^2 = 1 - \frac{1}{M} - \frac{MJ_n}{4}.$$

The last statement implies that

$$\max_{j=1,2,\dots,M} \epsilon_j \leq \sqrt{1 - \frac{1}{M} - \frac{MJ_n}{4}},$$

which is equivalent to

$$\min_{j=1,2,\dots,M} p_j = \frac{1}{M} - \max_j \epsilon_j \geq \frac{1}{M} - \sqrt{1 - \frac{1}{M} - \frac{MJ_n}{4}} = \frac{1}{M} - \frac{\sqrt{M(J^* - J_n)}}{2}.$$

□

Proof of Lemma 5. Since the split is perfectly balanced we have the following:

$$J_n = \frac{2}{M} \sum_{j=1}^M \sum_{i=1}^K q_i^{(n)} \left| \frac{1}{M} - p_{j|i}^{(n)} \right| = \frac{2}{M} \sum_{i=1}^K \sum_{j=1}^M q_i^{(n)} \left| \frac{1}{M} - p_{j|i}^{(n)} \right|$$

Define two sets

$$\mathcal{A}_i = \{j : j \in \{1, 2, \dots, K\}, p_{j|i}^{(n)} < \frac{1}{M}\} \quad \text{and} \quad \mathcal{B}_i = \{j : j \in \{1, 2, \dots, K\}, p_{j|i}^{(n)} \geq \frac{1}{M}\}.$$

Then

$$\begin{aligned} J_n &= \frac{2}{M} \sum_{i=1}^K \left[\sum_{j \in \mathcal{A}_i} q_i^{(n)} \left(\frac{1}{M} - p_{j|i}^{(n)} \right) + \sum_{j \in \mathcal{B}_i} q_i^{(n)} \left(p_{j|i}^{(n)} - \frac{1}{M} \right) \right] \\ &= \frac{2}{M} \sum_{i=1}^K q_i^{(n)} \left[\sum_{j \in \mathcal{A}_i} \left(\frac{1}{M} - p_{j|i}^{(n)} \right) + \sum_{j \in \mathcal{B}_i} \left(p_{j|i}^{(n)} - \frac{1}{M} \right) \right] \\ &= \frac{2}{M} \sum_{i=1}^K q_i^{(n)} \left[\sum_{j \in \mathcal{A}_i} \left(\frac{1}{M} - p_{j|i}^{(n)} \right) + \sum_{j \in \mathcal{B}_i} \left(\left(1 - \frac{1}{M}\right) - (1 - p_{j|i}^{(n)}) \right) \right] \end{aligned}$$

Recall that the optimal value of J_n is:

$$J^* = \frac{4}{M} \left(1 - \frac{1}{M}\right) = \frac{2}{M} \sum_{i=1}^N q_i^{(n)} \left[(M-1) \frac{1}{M} + \left(1 - \frac{1}{M}\right) \right] = \frac{2}{M} \sum_{i=1}^N q_i^{(n)} \left[\left(\sum_{j \in \mathcal{A}_i \cup \mathcal{B}_i} \frac{1}{M} \right) - \frac{1}{M} + \left(1 - \frac{1}{M}\right) \right]$$

Note \mathcal{A}_i can have at most $M-1$ elements. Furthermore, $\forall j \in \mathcal{A}_i, p_{j|i}^{(n)} < 1 - p_{j|i}^{(n)}$. Then, we have:

$$J^* - J^n = \frac{2}{M} \sum_{i=1}^K q_i^{(n)} \left[\sum_{j \in \mathcal{A}_i} p_{j|i}^{(n)} + \sum_{j \in \mathcal{B}_i} \left((1 - p_{j|i}^{(n)}) + \frac{1}{M} - (1 - \frac{1}{M}) \right) - \frac{1}{M} + \left(1 - \frac{1}{M}\right) \right]$$

Hence, since \mathcal{B}_i has at least one element:

$$\begin{aligned} J^* - J^n &\geq \frac{2}{M} \sum_{i=1}^K q_i^{(n)} \left[\sum_{j \in \mathcal{A}_i} p_{j|i}^{(n)} + \sum_{j \in \mathcal{B}_i} (1 - p_{j|i}^{(n)}) \right] \\ &\geq \frac{2}{M} \sum_{i=1}^K q_i^{(n)} \left[\sum_{j=1}^M \min(p_{j|i}^{(n)}, 1 - p_{j|i}^{(n)}) \right] \\ &\geq 2\alpha \end{aligned}$$

□

Proof of Theorem 1. Let the weight of the tree leaf be defined as the probability that a randomly chosen data point x drawn from some fixed target distribution \mathcal{P} reaches this leaf. Suppose at time step t , n is the heaviest leaf and has weight w . Consider splitting this leaf to M children n_1, n_2, \dots, n_M . Let the weight of the j^{th} child be denoted as w_j . Also for the ease of notation let p_j refer to $p_j^{(n)}$ (recall that $\sum_{j=1}^M p_j = 1$) and $p_{j|i}$ refer to $p_{j|i}^{(n)}$, and furthermore let q_i be the shorthand for $q_i^{(n)}$. Recall that $p_j = \sum_{i=1}^K q_i p_{j|i}$ and $\sum_{i=1}^K q_i = 1$. Notice that for any $j = \{1, 2, \dots, M\}$, $w_j = w p_j$. Let \mathbf{q} be the k -element vector with i^{th} entry equal to q_i . Define the following function: $\tilde{G}^e(\mathbf{q}) = \sum_{i=1}^K q_i \ln \left(\frac{1}{q_i} \right)$. Recall the expression for the entropy of tree leaves: $G^e = \sum_{l \in \mathcal{L}} w_l \sum_{i=1}^K q_i^{(l)} \ln \left(\frac{1}{q_i^{(l)}} \right)$, where \mathcal{L} is a set of all tree leaves. Before the split the contribution of node n to G^e was equal to $w \tilde{G}^e(\mathbf{q})$. Note that for any $j = \{1, 2, \dots, M\}$, $q_i^{(n_j)} = \frac{q_i p_{j|i}}{p_j}$ is the probability that a randomly chosen x drawn from \mathcal{P} has label i given that x reaches node n_j . For brevity, let $q_i^{n_j}$ be denoted as $q_{j,i}$. Let \mathbf{q}_j be the k -element vector with i^{th} entry equal to $q_{j,i}$. Notice that $\mathbf{q} = \sum_{j=1}^M p_j \mathbf{q}_j$. After the split the contribution of the same, now internal, node n changes to $w \sum_{j=1}^M p_j \tilde{G}^e(\mathbf{q}_j)$. We denote the difference between the contribution of node n to the value of the entropy-based objectives in times t and $t+1$ as

$$\Delta_t^e := G_t^e - G_{t+1}^e = w \left[\tilde{G}^e(\mathbf{q}) - \sum_{j=1}^M p_j \tilde{G}^e(\mathbf{q}_j) \right]. \quad (19)$$

The entropy function \tilde{G}^e is strongly concave with respect to l_1 -norm with modulus 1, thus we extend the inequality given by Equation 7 in [CCB16] by applying Theorem 5.2. from [AGNS11] and obtain the following

bound

$$\begin{aligned}
\Delta_t^e &= w \left[\tilde{G}^e(\mathbf{q}) - \sum_{j=1}^M p_j \tilde{G}^e(\mathbf{q}_j) \right] \\
&\geq w \frac{1}{2} \sum_{j=1}^M p_j \|q_j - \sum_{l=1}^M p_l q_l\|_1^2 \\
&= w \frac{1}{2} \sum_{j=1}^M p_j \left(\sum_{i=1}^K \left| \frac{q_i p_{j|i}}{p_j} - \sum_{l=1}^M p_l \frac{q_i p_{l|i}}{p_l} \right| \right)^2 \\
&= w \frac{1}{2} \sum_{j=1}^M p_j \left(\sum_{i=1}^K q_i \left| \frac{p_{j|i}}{p_j} - \sum_{l=1}^M p_{l|i} \right| \right)^2 \\
&= w \frac{1}{2} \sum_{j=1}^M p_j \left(\sum_{i=1}^K q_i \left| \frac{p_{j|i}}{p_j} - 1 \right| \right)^2 \\
&= w \frac{1}{2} \sum_{j=1}^M \frac{1}{p_j} \left(\sum_{i=1}^K q_i |p_{j|i} - p_j| \right)^2.
\end{aligned}$$

Before proceeding, we will bound each p_j . Note that by the *Weak Hypothesis Assumption* we have

$$\gamma \in \left[\frac{M}{2} \min_{j=1,2,\dots,M} p_j, 1 - \frac{M}{2} \min_{j=1,2,\dots,M} p_j \right],$$

thus

$$\min_{j=1,2,\dots,M} p_j \geq \frac{2\gamma}{M},$$

thus all p_j s are such that $p_j \geq \frac{2\gamma}{M}$. Thus

$$\max_{j=1,2,\dots,M} p_j \leq 1 - \frac{2\gamma}{M}(M-1) = \frac{M(1-2\gamma) + 2\gamma}{M}.$$

Thus all p_j s are such that $p_j \leq \frac{M(1-2\gamma) + 2\gamma}{M}$.

$$\begin{aligned}
\Delta_t^e &\geq w \frac{M^2}{2[(M(1-2\gamma) + 2\gamma)]} \sum_{j=1}^M \frac{1}{M} \left(\sum_{i=1}^K q_i |p_{j|i} - p_j| \right)^2 \\
&\geq w \frac{M^2}{2[(M(1-2\gamma) + 2\gamma)]} \left(\sum_{j=1}^M \frac{1}{M} \sum_{i=1}^K q_i |p_{j|i} - p_j| \right)^2 \\
&= w \frac{M^2}{8[(M(1-2\gamma) + 2\gamma)]} \left(\frac{2}{M} \sum_{j=1}^M \sum_{i=1}^K q_i |p_{j|i} - p_j| \right)^2 \\
&= \frac{M^2}{[(M(1-2\gamma) + 2\gamma)]} \frac{w J_n^2}{8},
\end{aligned}$$

where the last inequality is a consequence of Jensen's inequality. w can further be lower-bounded by noticing the following

$$G_t^e = \sum_{l \in \mathcal{L}} w_l \sum_{i=1}^K q_i^{(l)} \ln \left(\frac{1}{q_i^{(l)}} \right) \leq \sum_{l \in \mathcal{L}} w_l \ln K \leq w \ln K \sum_{l \in \mathcal{L}} 1 = [t(M-1) + 1] w \ln K \leq (t+1)(M-1) w \ln K,$$

where the first inequality results from the fact that uniform distribution maximizes the entropy.

This gives the lower-bound on Δ_t^e of the following form:

$$\Delta_t^e \geq \frac{M^2 G_t^e J_n^2}{8(t+1)[M(1-2\gamma)+2\gamma](M-1)\ln K},$$

and by using *Weak Hypothesis Assumption* we get

$$\Delta_t^e \geq \frac{M^2 G_t^e \gamma^2}{8(t+1)[M(1-2\gamma)+2\gamma](M-1)\ln K}$$

Following the recursion of the proof in Section 3.2 in [CCB16] (note that in our case $G_1^e \leq 2(M-1)\ln K$), we obtain that under the *Weak Hypothesis Assumption*, for any $\kappa \in [0, 2(M-1)\ln K]$, to obtain $G_t^e \leq \kappa$ it suffices to make

$$t \geq \left(\frac{2(M-1)\ln K}{\kappa} \right)^{\frac{16[M(1-2\gamma)+2\gamma](M-1)\ln K}{M^2 \log_2 e \gamma^2}}$$

splits. We next proceed to directly proving the error bound. Denote $w(l)$ to be the probability that a data point x reached leaf l . Recall that $q_i^{(l)}$ is the probability that the data point x corresponds to label i given that x reached l , i.e. $q_i^{(l)} = P(y(x) = i | x \text{ reached } l)$. Let the label assigned to the leaf be the majority label and thus let's assume that the leaf is assigned to label i if and only if the following is true $\forall_{z=\{1,2,\dots,K\}} q_i^{(l)} \geq q_z^{(l)}$. Therefore we can write that

$$\begin{aligned} \epsilon(\mathcal{T}) &= \sum_{i=1}^K P(t(x) = i, y(x) \neq i) \\ &= \sum_{l \in \mathcal{L}} w(l) \sum_{i=1}^K P(t(x) = i, y(x) \neq i | x \text{ reached } l) \\ &= \sum_{l \in \mathcal{L}} w(l) \sum_{i=1}^K P(y(x) \neq i | t(x) = i, x \text{ reached } l) P(t(x) = i | x \text{ reached } l) \\ &= \sum_{l \in \mathcal{L}} w(l) (1 - \max(q_1^{(l)}, q_2^{(l)}, \dots, q_K^{(l)})) \sum_{i=1}^K P(t(x) = i | x \text{ reached } l) \\ &= \sum_{l \in \mathcal{L}} w(l) (1 - \max(q_1^{(l)}, q_2^{(l)}, \dots, q_K^{(l)})) \end{aligned} \tag{20}$$

Consider again the Shannon entropy $G(\mathcal{T})$ of the leaves of tree \mathcal{T} that is defined as

$$G^e(\mathcal{T}) = \sum_{l \in \mathcal{L}} w(l) \sum_{i=1}^K q_i^{(l)} \log_2 \frac{1}{q_i^{(l)}}. \tag{22}$$

Let $i_l = \arg \max_{i=\{1,2,\dots,K\}} q_i^{(l)}$. Note that

$$\begin{aligned} G^e(\mathcal{T}) &= \sum_{l \in \mathcal{L}} w(l) \sum_{i=1}^K q_i^{(l)} \log_2 \frac{1}{q_i^{(l)}} \\ &\geq \sum_{l \in \mathcal{L}} w(l) \sum_{\substack{i=1 \\ i \neq i_l}}^K q_i^{(l)} \log_2 \frac{1}{q_i^{(l)}} \\ &\geq \sum_{l \in \mathcal{L}} w(l) \sum_{\substack{i=1 \\ i \neq i_l}}^K q_i^{(l)} \\ &= \sum_{l \in \mathcal{L}} w(l) (1 - \max(q_1^{(l)}, q_2^{(l)}, \dots, q_K^{(l)})) \\ &= \epsilon(\mathcal{T}), \end{aligned} \tag{23}$$

where the last inequality comes from the fact that $\forall_{i=\{1,2,\dots,K\}} q_i^{(l)} \leq 0.5$ and thus $\forall_{i=\{1,2,\dots,K\}} \frac{1}{q_i^{(l)}} \in [2; +\infty]$ and consequently $\forall_{i=\{1,2,\dots,K\}} \log_2 \frac{1}{q_i^{(l)}} \in [1; +\infty]$.

We next use the proof of Theorem 6 in [CCB16]. The proof modifies only slightly for our purposes and thus we only list these modifications below.

- Since we define the Shannon entropy through logarithm with base 2 instead of the natural logarithm, the right hand side of inequality (2.6) in [SS12] should have an additional multiplicative factor equal to $\frac{1}{\ln 2}$ and thus the right-hand side of the inequality stated in Lemma 14 has to have the same multiplicative factor.
- For the same reason as above, the right-hand side of the inequality in Lemma 9 should take logarithm with base 2 of k instead of the natural logarithm of k .

Propagating these changes in the proof of Theorem 6 results in the statement of Theorem 1. □

Proof of Corollary 1. Note that the lower-bound on Δ_t^e from the previous prove could be made tighter as follows:

$$\begin{aligned}
\Delta_t^e &\geq w \frac{1}{2} \sum_{j=1}^M \frac{1}{p_j} \left(\sum_{i=1}^K q_i |p_{j|i} - p_j| \right)^2 \\
&= w \frac{M^2}{2} \sum_{j=1}^M \frac{1}{M} \left(\sum_{i=1}^K q_i |p_{j|i} - p_j| \right)^2 \\
&\geq w \frac{M^2}{2} \left(\sum_{j=1}^M \frac{1}{M} \sum_{i=1}^K q_i |p_{j|i} - p_j| \right)^2 \\
&= w \frac{M^2}{8} \left(\frac{2}{M} \sum_{j=1}^M \sum_{i=1}^K q_i |p_{j|i} - p_j| \right)^2 \\
&= \frac{M^2 w J_n^2}{8},
\end{aligned}$$

where the first inequality was taken from the proof of Theorem 1 and the following equality follows from the fact that each node is balanced. By next following exactly the same steps as shown in the proof of Theorem 1 we obtain the corollary. □

Experimental Setting

In our experiments, we use a context window size of 3 for the PTB text, and 4 for Gutenberg. We optimize the objectives with Adagrad, and a learning rate of 0.025. We use L2 regularization for PTB with a multiplier of 5×10^{-5} , and no regularization for the Gutenberg data. The hidden representation dimension is 200 for both.